PILOT : Practical Privacy-Preserving Indoor Localization Using OuTsourcing

Järvinen, K.

IEEE
2019-06-17

Järvinen , K , Leppäkoski , H , Lohan , E , Richter , P , Schneider , T , Tkachenko , O & Yang , Z 2019 , PILOT : Practical Privacy-Preserving Indoor Localization Using OuTsourcing . in 2019 IEEE European Symposium on Security and Privacy (EuroS P) . IEEE , pp. 448-463 , 2019 IEEE European Symposium on Security and Privacy , Stockholm , Sweden , 17/06/2019 . https://doi.org/10.1109/EuroSP.2019.00040

http://hdl.handle.net/10138/309765
https://doi.org/10.1109/EuroSP.2019.00040

acceptedVersion

PILOT: Practical Privacy-Preserving Indoor Localization using OuTsourcing

Kimmo Järvinen
University of Helsinki
kimmo.u.jarvinen@helsinki.fi

Helena Leppäkoski
Tampere University
helena.leppakoski@tuni.fi

Elena-Simona Lohan
Tampere University
elena-simona.lohan@tuni.fi

Philipp Richter
Tampere University
philipp.richter@tuni.fi

Thomas Schneider
TU Darmstadt
schneider@encrypto.cs.tu-darmstadt.de

Oleksandr Tkachenko
TU Darmstadt
tkachenko@encrypto.cs.tu-darmstadt.de

Zheng Yang
Singapore University of Technology and Design
zheng_yang@sutd.edu.sg

***Abstract*—In the last decade, we observed a constantly growing number of Location-Based Services (LBSs) used in indoor environments, such as for targeted advertising in shopping malls or finding nearby friends. Although privacy-preserving LBSs were addressed in the literature, there was a lack of attention to the problem of enhancing privacy of indoor localization, i.e., the process of obtaining the users' locations indoors and, thus, a prerequisite for any indoor LBS.**

In this work we present PILOT, the first practically efficient solution for Privacy-Preserving Indoor Localization (PPIL) that was obtained by a synergy of the research areas indoor localization and applied cryptography. We design, implement, and evaluate protocols for Wi-Fi fingerprint-based PPIL that rely on 4 different distance metrics. To save energy and network bandwidth for the mobile end devices in PPIL, we securely outsource the computations to two non-colluding semi-honest parties. Our solution mixes different secure two-party computation protocols and we design size- and depth-optimized circuits for PPIL. We construct efficient circuit building blocks that are of independent interest: Single Instruction Multiple Data (SIMD) capable oblivious access to an array with low circuit depth and selection of the k-Nearest Neighbors with small circuit size. Additionally, we reduce Received Signal Strength (RSS) values from 8 bits to 4 bits without any significant accuracy reduction. Our most efficient PPIL protocol is 553x faster than that of Li et al. (INFOCOM'14) and 500x faster than that of Ziegeldorf et al. (WiSec'14). Our implementation on commodity hardware has practical run-times of less than 1 second even for the most accurate distance metrics that we consider, and it can process more than half a million PPIL queries per day.

I. Introduction

A Location-Based Service (LBS) is a service that relies on a user's physical location to provide the user with additional value. Pin-pointing the user's location on a map and offering routes to a given destination, navigation, targeted advertising, or finding nearby friends [1, 2, 3] are such add-on values that have already become a reality outdoors, but as well indoors. Privacy is a major concern in LBSs because the location history allows very accurate user profiling and even predicting users' future movements [4]. A lot of literature exists on privacy-preserving LBSs, e.g., [5, 6, 7, 8, 9]. Typically in these works, the user's location is assumed to be known by the user and the privacy concerns are about how this location information is shared and used in an LBS. In this paper, we focus on privacy issues of localization, i.e., of techniques for obtaining a user's location. We emphasize that privacy-preserving localization is the prerequisite for a privacy-preserving LBS because if privacy is violated already in the localization phase, then all further efforts for privacy protection are in vain.

Global Navigation Satellite System (GNSS) such as GPS or Galileo is the primary technology for localization in many LBSs [10]. Here, privacy of localization is not an issue because it is performed in a stand-alone GNSS receiver (e.g., in-car navigator) which calculates the location locally without any communication with other parties [11, 12]. Assisted or cloud GNSS receivers using cellular networks or the cloud for improving localization are exceptions where privacy of localization becomes an issue. Even though GNSS privacy is good, GNSS is unavailable or has poor services in indoor environments on which we focus in this work.

Thus, other means for localization must be used indoors. Proposed Indoor Localization (IL) techniques include Wi-Fi [13, 14, 15, 16], magnetic-field positioning, cellular, Bluetooth Low Energy (BLE) [17], and Ultra Wide Band (UWB) (see, e.g., [18, 19] for good surveys on indoor positioning technologies). Arguably, Wi-Fi fingerprinting is the most prominent and widely-adopted solution for IL [20, 21, 22, 23], as Wi-Fi networks and Wi-Fi capable devices are ubiquitous and therefore virtually free for the IL provider.

More recent Wi-Fi standards allow to measure additional parameters of the Wi-Fi signals such that more accurate propagation times [24] and angles of arrival [25, 26] can be obtained, enabling lateration or angulation localization methods. In multipath-free scenarios, where a signal can reach the user only once, these methods yield accuracies compareable with fingerprinting approaches. However, they do not preserve the users' privacy, require knowledge about the position of the Wi-Fi Access Points (APs), and the problems due to multipath propagation are not solved yet. Solutions that do not build upon Wi-Fi, such as cellular or UWB, either share these drawbacks or require additional network infrastructure. On the other hand, Wi-Fi fingerprinting is not limited to IL and can be used outdoors as well [27, 28].

Wi-Fi fingerprinting relies on the spatial distribution of the Received Signal Strength (RSS) of Wi-Fi APs. A user reads the RSSs of nearby APs and sends them to a server that hosts a database of pre-measured sets of RSS signatures and associated positions, so called Reference Points (RPs), encoding the measurement locations. The server then calculates the distances between the user's RSS measurements and the RSSs in the database under some distance metric. After that, the server finds the user's location as the RP associated to the RSSs that yielded the smallest distance. While Wi-Fi fingerprinting provides good localization accuracy and practical feasibility (e.g., cheap installation costs thanks to the use of the existing Wi-Fi infrastructure), there is the prevailing problem of privacy: the server, who calculates the user's location, can track the user's movements inside the building covered by IL. Also, the server has incentives not to publish its database and, hence, calculating locations locally in the user's device is typically not an option either as described next.

A. Motivation

Privacy-Preserving Indoor Localization (PPIL) is a very topical problem given the constantly increasing use of IL [29, 30, 31] in environments including shopping malls [21], exhibition centers, airports, hospitals, university campuses [22, 32], and museums. Examples of deployed IL services (mobile apps) include AnyPlace[1], HERE Indoor Positioning[2], IndoorAtlas[3], and Point Inside[4]. At the same time, with the constantly growing use of IL also the IL accuracy is constantly increasing, with recent papers pointing out to cm-level accuracy [30, 33]. Al Altogether, this means that the threats to the users' privacy are significant and growing [4, 11, 12, 34].

In the following, we consider the use of IL for customers of a shopping mall as a motivating example for PPIL. The customers of the shopping mall would greatly benefit from the availability of IL because it would help them to easily find shops, restaurants and other places of interest in the mall, and would also allow tailored flash discounts for nearby customers benefiting both the customers and retailers. However, the customers' movements in the mall and visits to specific places may reveal personal details and preferences that they are reluctant to share with anyone. Hence, privacy issues may hinder the adaptation of IL among privacy-aware customers. A trivial solution for PPIL would be to let the customers download local copies of the database, but the service provider has many incentives to keep the database secret. The database is a result of laborious measurements. An accurate database can be constructed only by making several measurements with different devices in exact locations, which typically requires special equipment, and/or time-consuming crowdsourced collection of fingerprints. Hence, the database is a central business secret of the LBS. Furthermore, the database may reveal sensitive information about the infrastructure (at least approximate locations of APs) and it is also not trivial to update a distributed database. Consequently, a PPIL scheme should prevent the leakage of both the customers' location and the LBS's database. Additionally, a PPIL scheme should permit extending the PPIL with arbitrary privacy-preserving functionalities (e.g., with privacy-preserving location-based marketing) in order to maintain the service provider's financial motivation to provide the service to its customers. Finally, a PPIL scheme should be efficient enough for practical deployment for large-scale use. In this paper, we describe PILOT, a PPIL scheme that fulfills the above requirements.

[1] https://anyplace.cs.ucy.ac.cy

[2] https://www.here.com/en/products-services/products/here-positioning/here-indoor-positioning

[3] http://www.indooratlas.com

[4] https://www.pointinside.com

B. Our Contributions

We provide the following contributions in this paper:

The First Practical PPIL Scheme. Our main contribution is the design and implementation of PILOT, the first Privacy-Preserving Indoor Localization (PPIL) scheme that (i) preserves the privacy of users' locations, (ii) protects the server's database, (iii) can be extended with arbitrary privacy-preserving functionalities such as location-based advertising, and (iv) is efficient enough for large-scale use in practice. PILOT is not just engineering, but a new synergy between the research areas of indoor localization and applied cryptography that leads to the *first* practically efficient and secure solution for PPIL. This main contribution is based on the following specific contributions.

Outsourcing Scenario and Impracticality of the Client/Server Scenario for PPIL. We use an efficient outsourcing protocol for computing PPIL using two Semi-Trusted Third Parties (STTPs). This protocol is very efficient for mobile clients, because it (i) increases the communication of the mobile client only by about factor $2\times$ compared to the non-private protocol, and (ii) enables cheap and energy-efficient computations for the mobile clients. In contrast, we show that even optimized PPIL in the setting with a single server using Secure Two-Party Computation (STPC) currently requires several hundred megabytes of communication and cryptographic operations which makes it impractical for mobile clients.

Quantization of the Received Signal Strength Values. To decrease the overhead of STPC evaluation, we quantize the RSS values with reduced bit-length and show the impact on the IL accuracy. As a result of a field experiment, we can conclude that the reduction from 8 to 4 bits has very limited impact on the accuracy of IL (0.1–5 %), and to 2 and 1 bit still achieves meaningful accuracy.

Well-Suited Distance Metrics. We design, implement, and evaluate a variety of privacy-preserving algorithms for computing conventional and more advanced distance metrics. We consider the Manhattan, Euclidean, Sørensen, and Kumar-Hassebrook distances, which differ in accuracy and efficiency.

Improved Building Blocks. Some of our newly introduced circuit building blocks are generic and thus of independent interest. We develop a circuit for

finding k-Nearest Neighbors (k-NN) out of N elements with $Nk(2\ell + \lceil \log_2 N \rceil)$ AND gates, and a Single Instruction Multiple Data (SIMD) capable oblivious access to an N-element array with $\lceil \log_2 \lceil \log_2 N \rceil \rceil + 1$ multiplicative depth.

Comprehensive Performance Analysis. We provide detailed benchmarking results for our best-performing and most precise algorithms for ranges of Reference Points (RPs) and Wi-Fi Access Points (APs). In addition, we provide even more detailed benchmarks for the same practical setting as in [35] with M=505 RPs and N=241 APs. Our most efficient algorithm has an online run-time of 0.15 s on two commodity servers which is orders of magnitude faster than prior work. Furthermore, the servers can process up to half a million PPIL queries per day.

II. Related Work

Many proposed solutions for Privacy-Preserving Indoor Localization (PPIL) use Homomorphic Encryption (HE) such as the Paillier cryptosystem [36]. Generic shortcomings of HE-based PPIL schemes are that they rely on computationally demanding Public-Key Encryption (PKE), and that a user would require to generate a new keypair at least for each visit of the building in order to prevent possible information leakage. Ideally, he or she would need to generate a new keypair for each query. Consequently, key generation and any setup phase precomputation would cause not only a large computational overhead, but also storage problems because of the very limited storage capacity in mobile devices. In contrast, our solutions perform efficient precomputation of the setup phase by the Semi-Trusted Third Parties (STTPs), i.e., only the online phase must be evaluated in real-time, and relies on symmetric-key cryptography. Moreover, the PPIL server shares its database only once in the STTP initialization phase. The client transfers as little as about twice the amount of communication of the currently used non-private protocol, it is not involved in the precomputation, and it performs only plaintext operations.

Li et al. [37] design algorithms for Privacy-Preserving Indoor Localization (PPIL) based on the k-Nearest Neighbors (k-NN) algorithm relying on Paillier encryption. Their work has several shortcomings: Yang and Järvinen [35] very recently presented an attack against this system that reveals the server's database under a realistic attack model. Our most efficient Euclidean distance-based PPIL protocol performs 121× faster with respect to the total run-time and 553× faster in the online phase than the protocol of [37] (see §VI-E for details).

Shu et al. [38] make use of the Paillier cryptosystem and Oblivious Transfer (OT) to construct a localization protocol that involves other mobile devices knowing their locations for determining the user's location. Our protocol requires only access to the mobile Internet by the mobile client. In [38], the weak battery-powered mobile client performs computational heavy public-key crypto, whereas in our protocol it performs only very cheap operations (Pseudorandom Generator (PRG) evaluations and XORs).

In [39], Ziegeldorf et al. apply HE for computing Hidden Markov Model-based PPIL. Our most efficient Euclidean distance-based PPIL protocol has 100× faster total run-time and 500× faster online run-time than the protocols in [39], even though we benchmark on a real mobile client and a cellular network, whereas they benchmark on two servers connected via a Gigabit LAN (see §VI-E for details).

Konstantidis et al. [40] use *k*-anonymity [41] for PPIL. In PPIL, the *k*-anonymity approach guarantees that the client is hard to identify in a set of *k* clients, and it generally reduces the utility of the data by adding noise.

Zhang et al. [42] propose Support Vector Machine (SVM)-based solutions for PPIL using HE. This is, however, insecure because the client can easily reconstruct the server's model using specially crafted queries. For example, the attack on a linear SVM with the kernel function $f(\vec{x}) = \vec{w} \cdot \vec{x} + b$ can be conducted as follows: (i) determine b by sending all-zero $\vec{x}$ and (ii) determine each element of $\vec{w}$ by sending a unit vector with a single one entry for each of the M elements in $\vec{x}$ with all other elements being zero.

Yang and Järvinen [35] present four high-level proposals for implementing secure PPIL: fully homomorphic encryption, Garbled Circuits (GCs), Paillier encryption only, and Paillier encryption combined with GCs. They conclude that while some of them are clearly unpractical (especially, fully homomorphic encryption), at least the one combining Paillier encryption with GCs should be feasible in practice. Their best scheme would require more than 3 MB communication per query (i.e., more than 1 GB per hour with 1 query every 10 s) vs. less than 2 kB per query in our work (less than 720 kB per hour with 1 query every 10 s). Unfortunately, the paper does not provide enough details to make a meaningful run-time comparison with our protocols possible. However, even their most efficient proposal requires expensive Paillier encryption operations, for computing Euclidean distances under encryption, and evaluation of a GC for k-NN computation (62 thousand AND gates in our setting), which introduce significant computational overheads for the server and the mobile client [43], and we expect their protocol to have run-times in the order of tens of seconds. Although the computation of a GC can be outsourced [44, 45], which decreases the overhead of the mobile client, it requires an additional party for the computation and hence they would be in the same setting as we are.

To summarize, the existing literature still lacks a PPIL solution that is both secure and efficient enough for practical deployment.

III. Preliminaries

In this section, we describe the distance metrics used for fingerprint-based Indoor Localization (IL) and the application of the k-Nearest Neighbors (k-NN) algorithm for IL. Afterwards, we give a summary of Secure Multi-Party Computation (SMPC). Table I depicts notation that we use throughout the paper.

Table I: Notation used throughout this paper.

P_0, P_1	Parties performing secure computation
$t \in \{A, B, Y\}$	Type of sharing: **A**rithmetic, **B**oolean, or **Y**ao
$\langle s \rangle_i^t$	Share s in sharing t held by party P_i
$\langle z \rangle^t = \langle x \rangle^t \odot \langle y \rangle^t$	Operation $\odot$ on shares $\langle x \rangle^t$ and $\langle y \rangle^t$
$\langle x \rangle^s = s2t(\langle x \rangle^t)$	Sharing conversion from source s to target t
$\langle 0 \rangle$, $\langle 1 \rangle$, $\langle 2 \rangle$	Secret-shared constants 0, 1, and 2
$\langle \mathbf{F}(\cdot) \rangle$	Secret-shared output of a local function **F**
$\mathbf{GT}(\langle x \rangle, \langle y \rangle)$	Greater Than gate yielding 1 if $x > y$, and 0 otherwise
$\mathbf{MUX}(\langle x \rangle, \langle y \rangle, \langle s \rangle)$	Multiplexer gate yielding $\langle x \rangle$ if $s = 0$, and $\langle y \rangle$ otherwise
$x \oplus y$ and $x \wedge y$	Bit-wise XOR and AND operation
$L[i]$	i-th element in list L
κ	Symmetric security parameter; κ=128 in this work

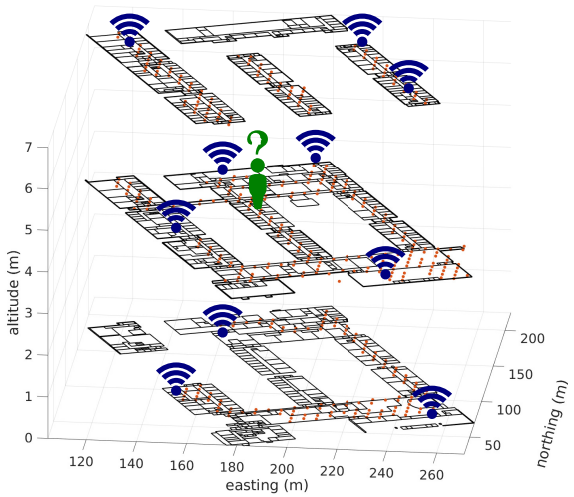

Figure 1: Exemplified setting for IL in a three-floor building with several Access Points (APs) (blue), pre-measured fingerprints at the Reference Points (RPs) (orange), and a client/user carrying a mobile device (green).

A. Indoor Localization

Location information already drives many services that are part of our daily routines. The spectrum of the services that rely on location information will broaden even further as the Internet of Things and 5G cellular networks emerge. The location information for these services is derived from the signal's propagation time, bearing or signal strength. We will focus here on Received Signal Strength (RSS)-based fingerprint localization, exemplified for Wi-Fi networks, as Wi-Fi fingerprinting is frequently used for IL due to its balance of complexity, availability of infrastructure, and accuracy [46]. However, our techniques can also be applied to fingerprint localization systems in other networks, e.g., cellular or Bluetooth networks. Fig. 1 presents a typical IL setting: a three-floor building with APs, RPs, and a user who wants to know his or her location.

Wi-Fi fingerprinting is a technique based on the spatial distribution of signal strengths. A location (RP) is assigned a signal strength signature, so-called fingerprint, that is ideally unique and therefore recognizable at a later point. It consists of the RSSs of the surrounding APs and identifiers relating the RSSs to their APs. To use Wi-Fi fingerprinting for localization, first, the service provider establishes an RSS map of the environment of interest. In a second step, a user records his or her RSS signature at a certain location, and the service provider can localize the user by comparing this RSS signature with the ones stored in the database holding the RSS map. The most likely location is the one whose fingerprint matches best the user's RSS signature under some distance metric, i.e., the Nearest Neighbor (NN) in RSS space.

Table II: Positioning performance of selected Received Signal Strength (RSS) distance metrics from [47] that are well-suited for Secure Multi-Party Computation (SMPC). The least accurate but most efficient metric is at the top, and the most accurate but least efficient metric is at the bottom.

Distance Metric	Success Rate (%)	Error (m)
Manhattan	90.73	7.06
Euclidean	92.71	7.40
Kumar-Hassebrook	94.33	7.00
Sørensen	94.78	6.86

The comparison of the user's RSS signature and the RSS signatures in the RSS map is done with distance measures which we detail below. The notation x_i and y_i represents the RSS of the i-th AP found in the RSS map and the one measured by the user, respectively. N is the number of APs. A comparison of the success and error rates using the different metrics is given in Tab. II. Success rate is the percentage of test points for which the building and the floor were estimated correctly. The error is the two-dimensional, horizontal distance between the estimated position and the ground truth position of test points; it was only computed for test points whose building and floor was estimated correctly.

1) Manhattan Distance: The Manhattan (also City Block) distance is a distance metric where only horizontal and vertical paths are allowed. It is calculated as shown in Eq. 1.

$$d_m = \sum_{i=1}^{N} |x_i - y_i| \tag{1}$$

2) Euclidean Distance: The Euclidean distance is the diagonal distance between two points. For computing the diagonal between two points, we apply the Pythagorean theorem to the distances between particular dimensions as shown in Eq. 2.

$$d_e = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2} \tag{2}$$

3) Sørensen Distance: The Sørensen distance is based on the Manhattan distance, which is normalized by the sum of the values. The calculation of this metric is shown in Eq. 3.

$$d_s = \frac{\sum_{i=1}^{N} |x_i - y_i|}{\sum_{i=1}^{N} (x_i + y_i)} \tag{3}$$

According to [47], the Sørensen Distance outperforms most other metrics for IL in terms of accuracy, and in particular the two aforementioned metrics (see Tab. II).

4) Kumar-Hassebrook Distance: The Kumar-Hassebrook distance is almost as accurate as the Sørensen distance, and it requires similar operations. It is computed as shown in Eq. 4.

$$d_{kh} = \frac{\sum_{i=1}^{N}(x_i \cdot y_i)}{\sum_{i=1}^{N} x_i^2 + \sum_{i=1}^{N} y_i^2 - \sum_{i=1}^{N}(x_i \cdot y_i)} \quad (4)$$

This distance metric performs better than the Euclidean and Manhattan distance, but marginally worse than the Sørensen distance (see Tab. II). However, as we will show in §V-B4, the Kumar-Hassebrook distance can be computed more efficiently in general-purpose SMPC than the Sørensen distance.

Choice of a distance metric: In this work, we aim at designing, implementing, and benchmarking a practically efficient PPIL system and PPIL protocols for different distance metrics. We select the distance metrics that either can be implemented very efficiently using SMPC or are one of the most accurate among others (see Tab. II) but can still be efficiently implemented. Based on the benchmarks, we show and discuss the trade-off between the more accurate but less efficient metrics and more efficient but less accurate metrics. However, the choice of the best-suited metric for a specific application is out of the scope of this paper.

Indoor Localization using k-Nearest Neighbors: To increase the accuracy of IL, the k-Nearest Neighbors (k-NN) algorithm can be used. In that case, not only one entry of the database is retrieved, but instead the positions corresponding to the k smallest RSS distances are averaged to estimate the user location. The value k is chosen to yield good performance [48]. Typical values are k=3 or k=4 [49, 50].

B. Secure Multi-Party Computation

A Secure Multi-Party Computation (SMPC) protocol enables n parties, each of whom holding a secret input, to compute a public function on their inputs without revealing anything except the result. In the following, we briefly review the main cryptographic tools and implementation of SMPC used in our solutions.

ABY Framework [51]: Among the variety of Secure Multi-Party Computation (SMPC) frameworks, e.g., [52, 51, 53, 54], we choose the ABY framework [51] for its efficiency and the ability to mix multiple SMPC protocols. ABY is a state-of-the-art framework for Secure Two-Party Computation (STPC). It supports three sharing types with state-of-the-art optimizations: Yao sharing based on Yao's Garbled Circuits (GCs) [55], Boolean sharing based on the GMW protocol [56], and Arithmetic sharing based on Arithmetic GMW. ABY guarantees passive security which means that an adversary is assumed to follow the protocol, but he or she may try to learn additional information from the received messages.

The STPC protocols have different benefits and drawbacks. GCs require a constant number of communication rounds, but need substantial communication and computation per AND gate. In low-latency networks, securely evaluating circuits with low depth using the GMW protocol is often faster than using GCs, but requires multiple rounds of interaction (one message for each layer of AND gates in the circuit). Arithmetic GMW allows computation of addition and subtraction without any interaction, and multiplication utilizes one Multiplication Triple (MT) [57] which can be efficiently precomputed using Oblivious Transfer (OT) extensions [58, 59] as shown in [51]. The drawback of Arithmetic GMW is that these are the only operations that can be computed efficiently.

The ABY framework allows to efficiently convert between these different types of sharing and benefit from their respective advantages. Moreover, it supports outsourcing of the computation to Semi-Trusted Third Parties (STTPs) and Single Instruction Multiple Data (SIMD) gates [60]. In principle, ABY also allows to compute very complex distance metrics that require logarithm or exponentiation operations using IEEE 754 Floating Point numbers [61], but this will slow down the protocols significantly and thus will not be considered in this paper.

In this work, we use Arithmetic sharing to improve the efficiency of distance computation, Boolean sharing to compute the depth-optimized Bitonic sort (for finding k nearest RPs), and Yao sharing to compute the size-optimized k-NN algorithm (instead of Bitonic sort). The choice between Boolean and Yao sharing depends on the network latency and the value of nearest neighbors k (see §VI).

Arithmetic Sharing: Arithmetic sharing is a protocol that enables two parties to secret-share an ℓ-bit integer x, and to evaluate a function which is expressed as an Arithmetic circuit consisting of addition and multiplication gates based on secret-shared values. Namely, for an integer x, each party has a random share $\langle x \rangle_i^A$, $i \in \{0,1\}$, such that $\langle x \rangle^A = \langle x \rangle_0^A + \langle x \rangle_1^A \pmod{2^\ell}$ and multiplication using the OT extension-based multiplication protocol of [51] requires O(ℓ) bits communication and one round of interaction, whereas addition and subtraction can be evaluated locally. However, XOR and AND operation are expensive in Arithmetic sharing as they would require expensive bit decomposition.

Boolean Sharing: In [56], Goldreich, Micali, and Wigderson (GMW) introduced a protocol which allows multiple parties to securely evaluate a function f expressed as Boolean circuit consisting of XOR and AND gates. We use the two-party variant of GMW, where each input and intermediate wire is shared by both parties with a 2-out-of-2 secret sharing scheme, i.e., each party P_i, $i \in \{0,1\}$, holds a Boolean share $\langle x \rangle_i^B$ for a secret-shared bit $\langle x \rangle^B$ such that $\langle x \rangle^B = \langle x \rangle_0^B \oplus \langle x \rangle_1^B$. While a XOR gate can be evaluated locally, an AND gate requires 2κ bits of communication in the setup phase and 4 bits in the online phase, as described in [59]. The communication of GMW can be further reduced at the cost of a higher computation complexity [62, 63].

Yao Sharing: Another approach for securely evaluating a Boolean circuit between two parties are Yao's Garbled Circuits (GCs) [55]. A GC can be generated as follows: the sender encrypts the Boolean gates of the circuit using randomly chosen symmetric keys for the wires and sends the encrypted function, called garbled circuit, together

with the keys representing its input bits to the receiver. The receiver obliviously obtains the keys corresponding to its inputs by running an Oblivious Transfer (OT) protocol with the sender which can be efficiently implemented using OT extension [58, 59]. Now, the receiver can evaluate the garbled circuit gate-by-gate using the keys and obtains the result. ABY implements state-of-the-art GC improvements such as point-and-permute [64], free-XOR [65], fixed-key AES garbling [66], and half-gates [67]. With this, XOR gates are free, whereas an AND gate costs 2κ bits of communication in the setup phase and no communication in the online phase, i.e., no additional rounds of communication.

Conversions: The cost of one Arithmetic to Yao (A2Y) or Arithmetic to Boolean (A2B) conversion is $6\ell\kappa$ bits of communication within 2 messages and 12ℓ AES operations, where ℓ is the bit-length of the Arithmetic share [51].

Hardware-Enhanced Secure Multi-Party Computation: Several works propose to use additional assumptions in form of trusted hardware to improve generic SMPC which could be used to improve PPIL schemes on mobile devices. Demmler et al. [68] use a smartcard in a mobile device for locally generating correlated randomness for the GMW protocol. Gupta et al. [69] design STPC protocols that use Intel SGX[5] for improving the efficiency of STPC. However, all these protocols require substantial trust in the hardware manufacturer.

IV. System Details

In this section, we describe our system PILOT for Privacy-Preserving Indoor Localization (PPIL) using outsourcing and detail the underlying security assumptions.

A. Outsourcing Scheme

Motivation: The best known solutions to PPIL without the non-colluding STTPs, i.e., PPIL in the client/single server setting, are very communication intensive. It is important to note that by using the outsourcing setting we achieve practical PPIL with security against malicious PPIL clients and malicious PPIL servers, whereas in the client/server setting even solutions with substantially weaker security against a semi-honest client currently have impractical runtimes and communication. Furthermore, the outsourcing model with multiple STTPs is widely adopted not only in recent academic papers, e.g., for private machine learning [70, 71], and genomic privacy [72, 73, 74, 75, 76, 77], to name just a few, but also deployed in industrial products, see [78] for a few examples.

Implementation: The outsourcing scheme we use is depicted in Fig. 2. The scenario involves three parties: a client who is interested in finding his or her coordinates, and two non-colluding Semi-Trusted Third Parties (STTPs), denoted as T_0 and T_1, that obliviously compute coordinates of the client using STPC. An additional entity in our scenario is the PPIL server (IL provider), which is not depicted in the figure because its overhead is one-time, and it is involved in the initialization of the STTPs only. On the initialization of the system or whenever updates occur (which in practice happens rarely), the PPIL server secret-shares its database of $N \cdot M$ Received Signal Strength (RSS) values between T_0 and T_1, where N is the number of APs and M is the number of RPs.

[5]https://software.intel.com/en-us/sgx

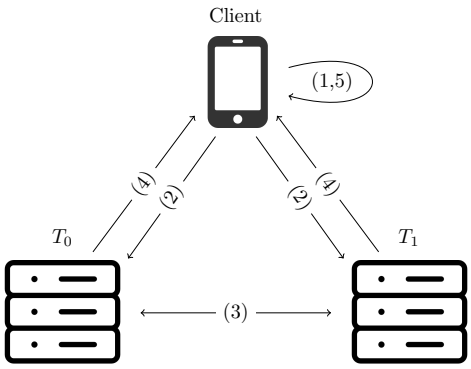

Figure 2: System model of our outsourcing scenario for efficient Privacy-Preserving Indoor Localization (PPIL). The PPIL server is not depicted in the figure, because it shares its database with the Semi-Trusted Third Parties T_0 and T_1 only once for the initialization of the protocol.

The STTPs are assumed to not collude. The assumption for STTPs not colluding is supported also by the facts that (i) a privacy-preserving service is a selling-point by itself to privacy-aware customers and getting caught from colluding would seriously damage reputation, and (ii) modern legislation, e.g., the EU General Data Protection Regulation (GDPR), emphasizes users' privacy and services have an obligation to protect their customers' data.

The PPIL works as follows (see Fig. 2):

1) The client collects RSS data from the APs (predefined by the PPIL server) to create a list $(\text{RSS}_1, \ldots, \text{RSS}_N)$, and secret-shares these values into $(\langle \text{RSS}_1 \rangle, \ldots, \langle \text{RSS}_N \rangle)$.
2) The client sends $(\langle \text{RSS}_1 \rangle_0, \ldots, \langle \text{RSS}_N \rangle_0)$ to T_0 and $(\langle \text{RSS}_1 \rangle_1, \ldots, \langle \text{RSS}_N \rangle_1)$ to T_1 over two secure channels.
3) The STTPs execute a PPIL protocol using STPC. As the result of the computation, they get the secret-shared coordinate $(\langle x \rangle, \langle y \rangle, \langle z \rangle)$.
4) T_0 sends $(\langle x \rangle_0, \langle y \rangle_0, \langle z \rangle_0)$ to the client and T_1 sends $(\langle x \rangle_1, \langle y \rangle_1, \langle z \rangle_1)$ to the client over two secure channels.
5) The client recombines the shares and reconstructs the cleartext coordinate (x, y, z).

B. Deployment

In a practical deployment, the parties in our system can be run by the following entities:

Client can be run by any visitor of the building on his/her mobile device.

Each of the STTPs can be operated by any of the following entities: the building owner, the mobile network operator, or a cloud provider.

PPIL server can be run by the mall or building owner. Real-world examples of IL providers are given in §I-A.

For the number of STTPs, we see three possible real-world settings: one, two, and three STTPs.

One STTP has the advantage that it requires only one server and one entity to operate the STTP. However, the computational overhead is substantial (cf. §II and §VI-E), and here the single server learns the database in the clear.

Two STTPs is the best model for real-world applications regarding the costs and security, since it is substantially more efficient than the model with one STTP, and the support and maintenance costs remain moderate compared to the setting with three STTPs. Furthermore, the STTPs obtain only a secret-shared database which preserves its privacy. Due to this, we choose the two-STTP setting in this paper.

Three STTPs can be more efficient than both previous models in an honest majority setting where at most one party can be corrupted, e.g., using the protocols of [79, 80]. However, it (i) guarantees a weaker level of security (it is easier to corrupt any two out of three parties than two out of two), and (ii) has much higher costs for the support and maintenance (to mitigate attacks, each STTP would need a different software stack and its own administration team).

C. Security Assumptions

In this work, we concentrate on a realistic scenario where the STTPs can only be passively corrupted, and the client and PPIL server can be actively corrupted. The latter is due to the fact that an actively corrupted PPIL server can only affect the correctness of the protocol by changing its inputs, and an actively corrupted client can only change the single message sent which corresponds to choosing a different input to the ideal functionality. The two STTPs have no inputs to the protocol (the inputs are already shared by the PPIL server and client) and thus can only try to learn more information from the protocol. Using standard techniques for outsourcing with active security [81, 82, 83], our protocols can be extended to full active security at the expense of higher communication and run-times. In this case, the system will detect one actively corrupted STTP, which can arbitrarily deviate from the protocol.

In order to guarantee privacy of the PPIL users, we make the following assumptions:

- When a device connects to a Wi-Fi Access Point (AP), it transmits to the AP its global Media Access Control (MAC) address, which is a unique, persistent identifier. Therefore, the client should be aware that there exists the possibility of being tracked by the AP owner, who could collude with the PPIL server or even be the same entity, e.g., the building owner provides a PPIL service and free Wi-Fi in the building. The use of such an AP would allow its owner to track the potential presence of the user in particular parts of the building. To mitigate this, we propose to use mobile Internet for PPIL, since even when a Wi-Fi enabled device is not connected to other Wi-Fi devices, it periodically scans its environment by sending probe requests. These requests also include the sender's MAC address and can be used to track the user [84]. However, in these requests the device is allowed to use a local MAC instead of its global MAC. Therefore, the client must use a randomized, changing local MAC address in the probe requests. This became possible in iOS version 8 and in Android version 6 [85].[6] One can root older Android devices or jailbreak older iOS devices that do not support randomized MACs by default, which enables randomized MACs and monitor mode.
- The STTPs must be chosen s.t. there is a strong incentive to not collude. One possibility is that the building operator operates T_0 and the user's mobile Internet provider operates T_1.
- For preventing an attacker from restoring the secret-shared inputs or coordinates by intercepting both shares, secure channels between the client and the STTPs must be used. However, a persistent TLS channel would leak information about the client. We avoid this potential leakage by establishing a new TLS connection for each new dynamically assigned IP address of the client. If the client's IP address is hidden behind a Network Address Translation (NAT) router getting a new network port for each query, he or she could establish a new TLS channel for each PPIL query to prevent tracking of his or her session.

D. Security Analysis

Here, we briefly analyze the security of our outsourcing scheme based on the SMPC model defined in [88] with concrete participants, i.e., client, PPIL server, and two outsourcing parties: STTPs T_0 and T_1. Kamara and Raykova [88] define and prove secure a generic construction that turns an n-party SMPC protocol into a protocol, where computation is outsourced to n non-colluding outsourcing parties. For our PPIL protocol, we instantiate the outsourcing scheme of [88] with the secure two-party (n=2) ABY framework. Our PPIL scheme is secure against semi-honest non-colluding STTPs, malicious clients, and malicious PPIL servers. Our threat model does not cover malicious Received Signal Strength (RSS) values and the leakage from the protocol output (i.e., client's coordinate), but in practice it would be hard to learn the PPIL server's database by fabricating RSS values.

Theorem IV.1. Suppose that the SMPC building blocks implemented in ABY [51] are secure, then our PPIL scheme is secure against passively corrupted non-colluding STTPs, and malicious clients and malicious PPIL servers.

Proof (sketch). Each secret-share $\langle \phi \rangle_i^t$ for $i \in \{0, 1\}$ and some value ϕ is blinded with a one-time random value (determined by the sharing type t of ABY). Namely, each uncompromised secret-share is statistically close to a truly

[6] Recent research, e.g., [86], have shown that MAC randomization is vulnerable to different types of attacks. As additional privacy measure, the client could set its Wi-Fi card into monitor mode which disables the probe requests but allows listening to the beacons from APs and measuring their RSS. In the current Android implementations, the Wi-Fi monitor mode requires rooting the Android device and installing the required application packages [87].

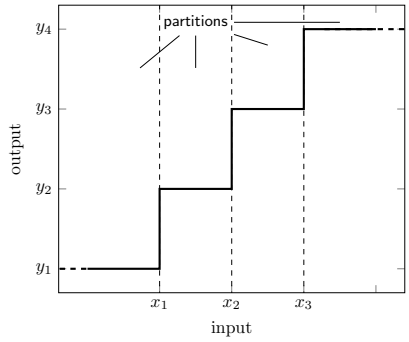

Figure 3: Uniform 2-bit quantizer showing the partitions.

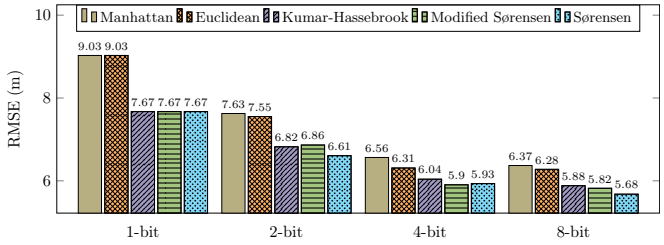

Figure 4: Positioning Root Mean Square Error (RMSE) in meters of k-NN for 1-, 2-, 4-, and 8-bit quantization of Received Signal Strength and different distance metrics obtained in a field experiment.

random value. Hence, a corrupted party obtaining a secret-share from its honest partner cannot obtain any useful information from it with overwhelming probability. Since two non-colluding STTPs cannot be both corrupted by definition, they will not collude in the security experiment. In this sense, at least one of the inputs of the subsequent STPC protocol is from an uncorrupted STTP (which received it from the corresponding uncorrupted party). Thus, we can reduce the security of our protocol to that of the STPC protocol (executed between STTPs) that uses the secure building blocks from the ABY framework. Namely, the security of the ABY-based STPC protocol ensures that the intermediate secret-shares (i.e., the outcome of any operation between secret shares), share conversions, and the final target location coordinate shares are secure as well. If the client and at least one STTP are not corrupted, then no party except the client can get the final location since the location coordinate shares are only reconstructed by the client. Analogously, if the PPIL server and at least one STTP are not corrupted, all secret shares leak no information about the RSSs of the server's database. PPIL server's malicious input can only affect the correctness of the protocol. Each malicious input from the client corresponds to a different input to the ideal functionality.

V. Privacy-Preserving Indoor Localization

In this section, we show our practical protocols for Privacy-Preserving Indoor Localization (PPIL) and optimizations.

A. Quantization of Received Signal Strength

We rely on the concept of a uniform quantizer to quantize the Received Signal Strength (RSS) values. A uniform quantizer is characterized by equally spaced boundary points [89] forming 2^ℓ non-overlapping partitions, where ℓ is the number of available bits. The midpoints of these partitions (except the two outermost partitions) correspond to the output levels, such that an input to the quantizer that lies within a partition is rounded off to the partition's midpoint. Each output level can be represented arbitrarily according to the application. Fig. 3 illustrates a uniform quantizer for ℓ=2.

To construct the quantizer for the RSSs, we specify the midpoints of the partitions by dividing the range between the lower and upper limit of the RSS values (of all fingerprints) by $2^\ell - 1$. To allow the localization algorithm to benefit from the information about the approximate coverage areas of the APs, we want to distinguish the cases when a signal is received with low RSS from the cases when a signal is not received at all. Therefore, we reserve one quantization partition for recording these signal-not-received cases and use this special partition during both the collection of the fingerprint database and the positioning phase. The boundary points for the RSS-quantizer are given by the midpoints plus half the size of a partition. The reserved partition for the zero-bit is added to the uniform quantizer, making it actually non-uniform. Its midpoint is chosen to be smaller than the lower RSS limit. For the purpose of RSS-based Indoor Localization (IL) via fingerprinting it suffices to represent the output levels of the quantizer simply by ℓ-bit codewords. The fingerprints of the Reference Points (RPs) can be quantized off-line on the server side. The fingerprints measured by the user must be quantized on the user device, thus, the quantizer must be made available for the clients.

Localization Performance: We compare the positioning accuracy, in terms of Root Mean Square Error (RMSE), of the k-NN algorithm for the four distance metrics specified in §III and consider additionally the modified Sørensen metric (see Eq. (5) in §V-B3). The data for that experiment was collected in a three-floor building at Tampere University of Technology, which is shown in Fig. 1. The corresponding database consists of M=446 RPs and N=449 APs; further details and the database itself are available at [90].

Fig. 4 shows the RMSE (averaged over a path through the building) for these four distances for several bit-lengths ℓ. The number of neighbors of the k-Nearest Neighbors (k-NN) was set to k=3. As shown in the figure, the accuracy of 4-bit quantization is similar to that of 8-bit with only 0.1–5 % loss of accuracy. Even with 1-bit quantization, a positioning accuracy of about 7–9 meters is achievable. This confirms the benefit of the information about the approximate coverage of the APs. The positioning error reduces with the number of bits spent to quantize the RSS, as expected. The Manhattan distance shows the poorest accuracy, whereas the (Modified) Sørensen distance is the most accurate one. We further investigate the effects of RSS quantization in [91].

Table III: AND-sizes and minimum bit-length of the circuits for the distance algorithms with N=241 Access Points, bit-length of Received Signal Strength ℓ, and the scaling factor of γ=16 bits.

Distance metric	Naive: Boolean / Yao Sharing	Optimized: Arithmetic, Boolean, Yao Sharing, OT				Min. bit-length	Min. bit-length with fixed ℓ			
	AND-size	AND-size	A$_*$	A2B / A2Y	C-OTs		ℓ=1	ℓ=2	ℓ=4	ℓ=8
Manhattan	$4(N \cdot \ell) - \ell$	$N \cdot (3\ell - \lceil \log_2 \ell \rceil - 2)$	0	0	$2N$	$\log_2 N + \ell$	8.9	9.9	11.9	15.9
Euclidean[7]	$N(2\ell^2 + 2\ell)$	0	N	0	0	$\log_2 N + 2\ell$	9.9	11.9	15.9	23.9
Sørensen	$6(N \cdot \ell) - 2\ell + 7\,079$	$N \cdot (3\ell - \lceil \log_2 \ell \rceil - 2) + 7\,079$	0	2	$2N$	$\log_2 N + \ell + \gamma$	24.9	25.9	27.9	31.9
Modified Sørensen	$N(2\ell^2 + 3\ell) + 7\,079$	7 079	N	2	0	$\log_2 N + 2\ell + \gamma$	25.9	27.9	31.9	39.9
Kumar-Hassebrook	$2N(3\ell^2 - \ell) + 2\ell + 7\,079$	7 079	N	2	0	$\log_2 N + 2\ell + \gamma$	25.9	27.9	31.9	39.9

B. Secure Multi-Party Computation of Distance Metrics

In this section, we describe in detail our privacy-preserving evaluation of the aforementioned distance metrics using the ABY framework [51]. These can be split into four categories: pure Arithmetic sharing (A), pure Boolean sharing (B), pure Yao sharing (Y), and mixed sharing (AB, AY, ABOY where O denotes OT) implementations. The latter use Arithmetic sharing first and then convert into Boolean or Yao sharing, respectively. In the following, we give the complexities for computing one distance, but for Privacy-Preserving Indoor Localization (PPIL), M instances need to be computed in parallel, one for each of the M Reference Points (RPs).

For Yao sharing we use size-optimized circuits of [92], and for Boolean sharing we use depth-optimized circuits of [60], as implemented in ABY [51], see [60, Tab. 7] for the complexities of the building blocks. We give an overview of size and depth complexities of the described algorithms as well as the minimum bit-length of the output to guarantee the correctness in Tab. III. In all our algorithms for securely computing distance metrics, we use sufficient bit-lengths which guarantees that the accuracy of the metrics does not decrease when compared with the cleartext protocols (denoted as minimum bit-length in Tab. III).

1) Manhattan Distance (cf. Eq. 1 on p. 4): The Manhattan distance requires computation of absolute values. This operation is nonlinear and hence cannot be efficiently computed in Arithmetic sharing. This is why we propose two workarounds to mitigate this issue: (i) enhancing the "naive" computation of the Manhattan distance without the loss of predicting accuracy by using Correlated OT (C-OT)-based multiplication of Arithmetic shares by a secret-shared bit (inspired by [93, 51]), and (ii) reducing the bit-length of the RSSs to 1 bit, which allows for the use of more efficient building blocks while guaranteeing accuracy acceptable in many scenarios.

To compute the Manhattan distance in a "naive" way, we use the size-optimized circuit of [92] that requires 1 Greater Than (GT), 2 Multiplexer (MUX), and 1 Subtraction (SUB) circuit which results in $4N\ell - \ell$ AND gates, where N is the number of APs and ℓ is the bit-length of the RSS values.

In our first improvement, we use very efficient C-OT-based multiplication of secret-shared values to choose either the initial result of $\langle \mathrm{RSS}_C \rangle^A - \langle \mathrm{RSS}_S \rangle^A$ or its two's complement [94] based on the outcome of the comparison of both in Boolean sharing (the comparison can also be performed by using the underflow bit of the result, but this also incurs some overhead because of the conversion to Boolean sharing). This does not violate the security, since the multiplication protocol is performed obliviously on random-looking secret-shared values and yields random-looking secret-shared values as the result. A similar technique was used, for example, in the ABY framework for precomputing MTs [51]. This protocol is not only much more efficient than doing this using the "naive" solution, but also allows to sum up the distances of all dimensions locally, since the computation yields Arithmetic shares for each dimension. This protocol requires only 1 Greater Than (GT) gate and 2 C-OTs for each dimensions resulting in $N(3\ell - \lceil \log_2 \ell \rceil - 2)$ AND gates and $2N$ parallel C-OTs of ℓ-bit strings. We depict our C-OT-based implementation of the Manhattan distance in Alg. 1 in §A.

In our second improvement, which we denote as 1-bit Manhattan distance, we use 1-bit RSS values and compute the distance in pure Yao sharing. Since the Manhattan distance of 1-bit values corresponds to their XOR, we perform the computation non-interactively. To compute the sum of the computed distances, we utilize the AND-optimized Hamming Weight (HW) circuit of Boyar and Peralta [95] with $\ell - d_H(\ell)$ AND gates, where $d_H(\ell)$ denotes the HW of ℓ. This protocol comes at the price of accuracy reduction, which, however, remains meaningful for many scenarios, i.e., 9 m RMSE in our experiments (see Fig. 4). By replacing the HW circuit with the custom C-OT-based protocol of [96] for securely computing an Arithmetic sharing of the HW, the communication of this protocol can be improved by a factor of 3.7× from 5.2 MB to 1.4 MB. Overall, this improves communication over the naive approach by factor 215×.

2) Euclidean Distance (cf. Eq. 2 on p. 4): Since computing the square root is very expensive in SMPC, we replace the Euclidean distance with its squared version, which only omits the last operation of computing the square root. Note that this does not affect the outcome of finding the nearest RPs because all distances are squared. The Boolean circuit-based Euclidean distance requires $N(2\ell^2 + 2\ell)$ AND gates, whereas the Arithmetic sharing optimized algorithm can completely be computed using Arithmetic sharing only. This algorithm utilizes N multiplication gates in Arithmetic sharing. An important difference to the other securely computed distance metrics is that the result of the computation has to be converted to another sharing type for efficient computation of k nearest RPs, which for other metrics either happens during the distance computation (if required) or does not happen at all (if not required). We give our Arithmetic sharing-based implementation of the Euclidean distance in Alg. 2 in §A.

3) Sørensen Distance (cf. Eq. 3 on p. 4): The Boolean circuit-based Sørensen Distance can be computed using $6(N \cdot \ell) - 2\ell + 7\,079$ AND gates. To reduce the overhead for computing the Manhattan distance in the numerator, we utilize the idea of the C-OT-based Manhattan distance protocol. Since the addition of the values in the denominator is performed on the Arithmetic shares, the difference in the interactive operations to the Manhattance distance protocol comes from the 32-bit division operation, which costs 7079 AND gates.

Modified Sørensen Distance: Since the computation of absolute values cannot be efficiently done in Arithmetic sharing, we introduce a more SMPC-friendly variant of the Sørensen Distance where we replace the Manhattan distance by the Euclidean distance (cf. Eq. 5) and evaluate its IL accuracy. We denote this distance metric as modified Sørensen distance.

$$d'_s = \frac{\sum_{i=1}^{D}(x_i - y_i)^2}{\sum_{i=1}^{D}(x_i + y_i)} \tag{5}$$

The Root Mean Square Error (RMSE) of the modified Sørensen distance is depicted in Fig. 4, and in particular settings this metric is even more accurate than the original Sørensen distance. The Boolean circuit-based Modified Sørensen distance can be computed using $N(2\ell^2+3\ell)+7\,079$ AND gates. As a result, this protocol requires 7 079 AND gates, N multiplication gates in Arithmetic sharing, and 2 A2B or A2Y conversions, respectively.

In Alg. 4 in §A, we show the AY implementation of the modified Sørensen distance. We use a γ=16-bit scaling factor for both Sørensen and Kumar-Hassebrook distance to achieve a precision comparable to that of floating point arithmetic, i.e., we perform a bit-shift on the distance by γ bits before dividing it by the denominator.

4) Kumar-Hassebrook Distance (cf. Eq. 4 on p. 5): The Kumar-Hassebrook distance, unlike the Sørensen distance, requires multiplications in the numerator. Therefore, its Boolean circuit-based algorithm requires $2N(3\ell^2 - \ell) + 2\ell + 7\,079$ AND gates. However, it can naturally be optimized in Arithmetic sharing. The complexities of the optimized algorithm are as follows: 7 079 AND gates, N multiplication gates in Arithmetic sharing, and two A2B or A2Y conversions, respectively. The implementation of the optimized Kumar-Hassebrook distance is shown in Alg. 3 in §A.

C. k-Nearest Neighbors

Songhori et al. [97] propose a circuit for k-Nearest Neighbors (k-NN) with size $M(2k-1)(2\ell+\lceil\log_2 M\rceil)$ AND gates, where M is the number of elements (here, the number of Reference Points) and ℓ is their bit-length.

Optimized k-Nearest Neighbors: We improve the k-NN circuit of [97] by using more efficient conditional swap circuits of [65]. This results in $Mk(2\ell + \lceil\log_2 M\rceil)$ AND gates which for k=3 is an improvement by factor 1.6× over [97]. Our modified k-NN implementation is given in Alg. 6 in §A.

k-Nearest Neighbors via Oblivious Sorting: An alternative approach for computing k-NN is to use Oblivious Sorting. This improves the efficiency of the non-constant round SMPC protocols such as GMW because the k-NN algorithm of [97] has high depth of $\mathrm{O}(Mk\ell)$. Instead, we use the Bitonic Sort algorithm of [98] that is summarized in Alg. 5 in §A. As a baseline for our implementation, we use the algorithm of [99]. The algorithm requires $0.5 \cdot \lceil\log_2 M\rceil(\lceil\log_2 M\rceil + 1)$ stages with $M(4\ell-\lceil\log_2 \ell\rceil-2+\lceil\log_2 M\rceil)$ AND gates each. This is by factor ∼12x larger than for our optimized k-NN algorithm for M=505 and k=3. However, this algorithm produces a Boolean circuit that is by factor ∼10x shallower than that of our improved k-NN already for k=3, and its AND-depth does not grow with k.

D. Oblivious Array Access

In order to find the user's coordinates in Privacy-Preserving Indoor Localization (PPIL), we have to find the coordinates that correspond to the nearest Reference Points (RPs) by their secret-shared indices computed by the k-NN algorithm. We use an approach similar to the one proposed by Keller and Scholl [100] for our size-optimized implementation, and design a new Single Instruction Multiple Data (SIMD) capable circuit for the depth-optimized implementation with the same AND-depth and slightly larger AND-size than the circuit proposed by Buescher et al. [101] (the latter cannot be parallelized easily with SIMD).

The size-optimized circuit is built as an inverted binary tree, i.e., initial elements represent leaves, and the selected element represents the root. Each intermediate node between the root and the leaves is a MUX-gate which has two elements and a choice bit as input. The tree has $\lceil\log_2 M\rceil$ depth, where M is the length of the array, and all MUX-gates on the layer i have the same choice bit $\langle idx\rangle^b[i]$, i.e., the i-th bit of the secret-shared index $\langle idx\rangle^b$ with $b \in \{B, Y\}$. This requires $M-1$ MUX-gates which yields $\ell(M-1)$ AND gates (8 064 for M=505 and ℓ=16), where ℓ is the bit-length of the coordinate. The AND-depth is hereby $\lceil\log_2 M\rceil$ (9 for M=505). The size-optimized circuit is depicted in Alg. 7 in §A.

The depth-optimized circuit is designed as follows: the server provides a list l which contains the secret-shared coordinates corresponding to the APs. For each index $\langle idx\rangle^b$ which corresponds to one of the k nearest APs, we compare $\langle idx\rangle^b$ with $(\langle 1\rangle^b, \ldots, \langle M\rangle^b)$ using a comparison circuit. This yields a list of secret-shared zeros with a single secret-shared one entry. Afterwards, we apply a MUX gate to each set of coordinates which results in all coordinates being zeros except of exactly one set. To get the final result, we perform the XOR operation on all the coordinates. This circuit has AND-depth $\lceil\log_2\lceil\log_2 M\rceil\rceil + 1$ (5 for M=505, 44 % shallower than the size-optimized) and $M(\lceil\log_2 M\rceil - 1 + \ell)$ AND-size (12 120 for M=505 and ℓ=16, by 50 % larger than the size-optimized). The depth-optimized circuit is shown in Alg. 8 in §A. In contrast to the circuits of Buescher et al. [101],

[7]The optimized Euclidean distance uses only Arithmetic sharing.

our Oblivious Array Access (OA) circuit is slightly larger, but it has a much simpler structure and therefore the same sub-circuits can be applied in parallel using SIMD.

For PPIL, the circuits are applied kD times to find all D dimensions of the k coordinates. Note that for both Oblivious Array Access circuits the AND-depth is independent of k and D because the circuits are evaluated in parallel.

E. Extended Functionalities

A service provider runs the Indoor Localization (IL) service in order to get financial profit, e.g., by using customers' location information for location-based advertising. Hence, it is essential that Privacy-Preserving Indoor Localization (PPIL) can be extended with other Location-Based Services (LBSs), because otherwise the service provider could lose its incentive to offer the IL service in the first place. We emphasize that this requirement is met by PILOT. The extended functionalities can be implemented either (i) as an additional layer within PILOT (which would even not leak the location as intermediate result) or (ii) as an additional privacy-preserving protocol after localization with PILOT (which leaks the location).

VI. Evaluation

We implement and benchmark our protocols on two servers acting as Semi-Trusted Third Parties (STTPs) T_0 and T_1, respectively, each equipped with an Intel Core i7-4770K 3.5 GHz processor and 16 GB RAM. The servers are located in a local Gigabit network with average network latency of about 0.1 ms. The client is run on a Samsung Galaxy S7 mobile device equipped with a Qualcomm MSM8996 Snapdragon 820 processor and 4 GB RAM and is connected to the servers via LTE mobile Internet with 229 ms average Round Trip Time (RTT) and 163 ms median RTT.

Here, we describe the benchmarking results of our algorithms in the outsourcing scenario. We instantiate all primitives with a security level of κ=128-bit and report run-times averaged over 10 executions. We use 4-bit Received Signal Strength (RSS) values (see §V-A) in all but one (1-bit Yao sharing-based Manhattan distance) of our benchmarks.

For establishing a TLS 1.2[8] connection between the client and the STTPs given an established TCP connection (establishing a TCP connection adds a delay of $\sim$1 RTT), we measured 468 ms average run-time, which is approximately equal to 2 RTTs. The time for transferring the data was approximately 1 RTT. These additional time demands depend almost only on the network latency which significantly varies depending on the environment. Note that with broader use of TLS 1.3[9], the latency for establishing a secure channel will be reduced by half due to the 1-RTT handshake protocol. Furthermore, in TLS 1.3 it is possible to resume the connection at very low additional costs due to the 0-RTT connection resumption protocol.

[8]https://tools.ietf.org/html/rfc5246
[9]https://tools.ietf.org/html/rfc8446

A. Input Sharing of Mobile Client

As shown in Tab. IV, the overhead for sharing the RSS values ranges from 0.2 ms to 6.1 ms depending on the number of Access Points (APs) N and hence is negligible compared to the run-time of the STPC protocols, e.g., about 2 % overhead compared to the run-time in the online phase of the most efficient Euclidean distance-based PPIL. In addition, both the bit-length of the shared value and the sharing type only slightly influence the run-times because of the high level of abstraction for the Java code that runs on the mobile client. The shares are created using a modified `java.security.SecureRandom` class, which is extended for generating also values of small byte-lengths.

B. Precomputation in the Setup Phase

As shown in Tab. V, the setup phase of the 4-bit protocols has between 164 MB and 6.1 GB communication and a run-time between 0.7 s and 66.3 s, depending on which distance metric and protocol is used. Since the setup phase is independent of the inputs (only correlated randomness is produced), it can efficiently be precomputed by the STTPs while they are idling, e.g., over night. The most important advantage of our outsourcing scheme is that this setup phase is run only between the STTPs, i.e., it is independent of the PPIL server and client. Moreover, the PPIL computation is independent of the user's identity, i.e., neither the identity of the user influences the setup phase nor the precomputation of the setup phase violates the privacy of the user. Due to the fact that the STTPs are easy to equip with hard disk drives of large capacity for a reasonable price, the STTP servers can precompute thousands of PPIL queries. If this throughput is not sufficient, we can think of the following: one can either upgrade the available hardware or continue the precomputation on the days while the building is closed (weekends, holidays). While the first gives many degrees of freedom for possible efficiency improvements, the second allows the precomputation of more than 100 000 queries during only one weekend.

C. k-Nearest Reference Points

We show the performance differences between the k-NN and Bitonic Sort algorithm on the left of Tab. IV. As it can be seen, k-NN in Yao sharing significantly outperforms Bitonic Sort in Boolean sharing for k=3. On the other hand, Bitonic Sort becomes more efficient with increasing k and would be more efficient than k-NN when k is greater than 40 or 50 because its performance is independent of k. However, for IL k is usually small, e.g., k=3 or k=4. Our k-NN protocol achieves a $1.4\times$ to $1.6\times$ run-time improvement over [97].

D. Oblivious Array Access

The run-times of the Yao-based size-optimized and GMW-based depth-optimized Oblivious Array Access (OA) algorithm for the ranges of RPs are given in the middle of Tab. IV. As it turns out, the size-optimized implementation of OA slightly outperforms the depth-optimized implementation in our standard setting with M=505 RPs. While the first

Table IV: The run-times in milliseconds for sharing client's inputs and for the online phase of our most promising PPIL protocols for N Access Points (APs) and M Reference Points (RPs). In the right part of the table, we depict the best-performing protocols for PPIL using the corresponding distance metrics. k-Nearest Neighbors (k-NN) is implemented in Yao and Bitonic Sort in Boolean sharing.

M	Input sharing	k nearest RPs				Oblivious Array		Manhattan (ABOY)		Euclidean (AY)		Kumar-Hassebrook (AY)		Sørensen (ABOY)		Mod. Sørensen (AY)	
		3-NN	3-NN [97]	30-NN	Bitonic Sort	Size-opt.	Depth-opt.	N=256	N=512	N=256	N=512	N=256	N=512	N=256	N=512	N=256	N=512
32	0.2	0.7	1.1	8.3	26.5	1.2	2.1	47.2	75.0	11.7	18.9	57.6	61.1	66.2	80.9	58.8	61.6
64	0.4	1.5	2.3	16.3	51.9	2.3	4.1	85.6	151.9	20.6	35.8	109.7	114.5	123.0	151.2	112.0	119.3
128	0.8	3.1	4.5	32.7	106.1	4.6	7.3	159.8	289.7	38.3	62.0	213.6	230.5	234.9	293.6	215.4	228.9
256	1.5	5.9	10.3	59.0	186.9	9.3	15.6	304.1	570.9	74.5	124.1	420.3	455.2	476.7	575.0	424.9	455.2
512	3.1	12.3	19.7	120.4	332.4	20.1	31.0	599.7	1 137.4	162.5	270.9	843.8	922.8	948.7	1 144.4	874.6	948.1
1 024	6.2	24.7	40.4	239.8	681.3	39.5	63.8	1 184.6	2 167.8	363.7	594.3	1 646.4	1 969.4	1 873.4	2 179.8	1 710.1	1 979.9

Table V: Benchmarking results for Privacy-Preserving Indoor Localization run between two Semi-Trusted Third Parties (STTPs) based on the introduced distance metrics with N=241 Access Points and M=505 Reference Points. The most efficient solutions for each metric are marked in bold.

Distance metric (Protocol)	Comm. across STTPs in MB			Runtimes across STTPs in s		
	Setup	Online	Total	Setup	Online	Total
Manhattan (B)	974	15	990	4.2	15.0	20.6
Manhattan (Y)	301	51 B	301	3.9	2.2	6.1
Manhattan (ABOY)	167	11	178	0.7	0.4	1.1
Manhattan (1-bit, Y)	5.2	**47 B**	**5.2**	0.3	**0.2**	**0.5**
Euclidean (Y)	6 100	**50 B**	6 100	66.3	16.9	82.3
Euclidean (AB)	203	4.0	207	1.0	0.9	1.9
Euclidean (AY)	164	2.7	**167**	0.8	**0.15**	**1.0**
Kumar-Hassebrook (AB)	319	6.6	326	1.6	2.5	4.1
Kumar-Hassebrook (AY)	280	**2.5**	**283**	2.2	**0.8**	**3.0**
Sørensen (ABO)	196	9.2	205	1.1	2.0	3.1
Sørensen (ABOY)	148	**5.7**	**154**	1.5	**0.9**	**2.6**
Modified Sørensen (AB)	321	6.6	328	1.6	2.4	4.0
Modified Sørensen (AY)	282	**2.6**	**285**	2.2	**0.8**	**3.0**

requires 34 ms run-time in the online phase, the second runs in 39 ms which is by $\sim$15 % slower. However, in total the depth-optimized algorithm runs by factor $\sim$1.5$\times$ faster than the size-optimized (46 ms vs. 72 ms) and requires almost 2$\times$ less communication (1.3 MB vs. 2.3 MB).

E. Performance of Privacy-Preserving Indoor Localization

Concrete parameters for Indoor Localization (IL) depend on the specific application:

- The accuracy of IL depends on the number of RPs, the demands of which depend on the particular scenario, e.g., in one scenario we need to determine whether a user is in a room and in another scenario whether the user is in a particular part of the room.
- One will not always require all possible RPs in the building, e.g., only one floor, or all floors, but only one wing of the building.
- The APs often have many MAC addresses which causes duplicating the same APs in IL. Duplicates can be filtered out by creating the public list of single APs. Moreover, the APs that provide unreliable data can also be filtered out using machine learning techniques, such as [102]. Using these techniques, also the most reliable APs can be found.

In this section, we present the performance results for our PPIL protocols: (i) for a real-world dataset with N=241 APs and M=505 RPs, (ii) for the settings used in the existing literature for comparing our solutions with those proposed in the previous works, and (iii) for ranges of APs and RPs.

Real-World Setting: In Tab. V we show the benchmarking results of our algorithms for N=241 APs, M=505 RPs, and k=3 Nearest Neighbors. The setting is the same as in [35] and represents a typical size for a single building database. Examples on databases with similar sizes can be found, e.g., in [103, 104, 105]. Note that only algorithms fitting into 16 GB RAM on the STTPs are depicted in the table. Since the setup phase can efficiently be precomputed, the most important information in this table are the online run-times and communication.

We can deduce the following from Tab. V:

- Even with all our optimizations, 4-bit PPIL in the client/server setting with a single server would require transferring at least 100 MB between the mobile client and the server over the mobile Internet per query. This is highly unpractical, especially given today's mobile dataplans of usually a few GB per month. Therefore, outsourcing to two or more non-colluding servers as in the model investigated in this work is the only realistic model because only a few kB need to be sent over the mobile Internet.
- Our most efficient Euclidean distance-based PPIL protocol using Arithmetic and Yao sharing achieves 1.0 s total and 0.15 s online time, which is a reasonable delay in practice.
- Many optimized algorithms require less than 1 second online time even for the most precise metrics which is also a reasonable delay in practice.
- The optimized algorithms outperform the non-optimized ones in terms of online run-time by up to factor 100$\times$.
- The optimized algorithms in Yao sharing outperform the ones in Boolean sharing in terms of online time by factor 3–4$\times$. This is due to the fact that although the circuits in Boolean sharing have reasonable depth (298–596), they are up to factor 6$\times$ larger than the circuits in Yao sharing.
- For the Manhattan/Euclidean/Kumar-Hassebrook/Sørensen/Modified Sørensen-based PPIL in Arithmetic and Yao sharing (AY), our two commodity hardware-based servers can daily precompute 172k/86k/28k/33k/28k queries, and process 432k/576k/108k/96k/108k queries, respectively.
- The 1-bit Manhattan distance PPIL protocol yields 57$\times$ less communication than the "naive" solution (only 5.2 MB in total), it requires only 0.5 s of total run-time, 0.2 s online time, and its online communication is in the order of a few Bytes, which makes this protocol an excellent candidate also for the client-server scenario without outsourcing.

Comparison with Related Work: In the following, we compare our most efficient 4-bit Euclidean distance-based PPIL protocol with the most recent related works of Li et al. [37] and Ziegeldorf et al. [39].

[37]: The experiments of [37] have very high run-times of 60.9 s for very small IL parameters (N=15 APs and M=1 000 RPs) and outdated security parameters (1024-bit Paillier modulus which is considered insecure today [106]). For these values of N and M and state-of-the-art 128-bit security, our solution has an online run-time of 0.11 s (553× faster) and a total run-time of 0.5 s (121× faster).

[39]: Evaluation of the prototype implementation of [39] on two servers with 1 ms network latency resulted in a 10 s run-time for only N=20 APs and 160 possible states. For the same setting (we set the number of RPs to M=160), we achieve an online run-time of 0.02 s (500× faster) and a total run-time of 0.1 s (100× faster).

Ranges of APs and RPs: An intuitive solution for representing the real-world scalability of our algorithms is to perform benchmarks for different real-world buildings. This, however, has the drawback that the benchmarks for a few buildings are valid only for these buildings and say nothing about other buildings. To solve this problem, we benchmark our algorithms on ranges of APs and RPs in constant intervals. This allows to interpolate the benchmarking results for determining overhead for any numbers of APs and RPs in the measured range. Since the function or shape of a building does not affect the performance of our algorithms, the benchmarking results will apply to all buildings with the same number of APs and RPs. We give the online times for our most efficient protocols for ranges of APs and RPs in Tab. IV.

VII. Conclusion

In this work, we designed, implemented, and evaluated PILOT, a system for practical Privacy-Preserving Indoor Localization (PPIL). To the best of our knowledge, PILOT is the first PPIL system with practical online run-times of less than 1 s and is by several orders of magnitude faster than previous works. We applied quantization to reduce the Received Signal Strength bit-length without significant loss of accuracy. We designed efficient building blocks for PPIL, which are of independent interest: a size-efficient k-Nearest Neighbors circuit, and a Single Instruction Multiple Data-capable and easy-to-implement Oblivious Array Access circuit. We used an outsourcing scenario that shifts most of the communication and computation away from the weak and battery-powered mobile client. Overall, our PPIL system PILOT can process hundreds of thousands of queries per day on commodity hardware.

Acknowledgements

This work was supported by the German Federal Ministry of Education and Research (BMBF) and by the Hessen State Ministry for Higher Education, Research and the Arts (HMWK) within CRISP, by the DFG as part of project E4 within the CRC 1119 CROSSING, by the Academy of Finland under grant number 303576 and 303578, and by the National Natural Science Foundation of China under Grant No. 61872051.

References

[1] A. Kushki, K. Plataniotis, and A. Venetsanopoulos, *WLAN positioning systems*. Cambridge University Press, 2012.

[2] S. Alletto, R. Cucchiara, G. D. Fiore, L. Mainetti, V. Mighali, L. Patrono, and G. Serra, "An indoor location-aware system for an IoT-based smart museum," *IEEE IoT-J*, 2016.

[3] R. Li, T. Song, N. Capurso, J. Yu, J. Couture, and X. Cheng, "IoT applications on secure smart shopping system," *IEEE IoT-J*, 2017.

[4] S. M. Bellovin, R. M. Hutchins, T. Jebara, and S. Zimmeck, "When enough is enough: Location tracking, mosaic theory, and machine learning," *NYU Journal of Law & Liberty*, 2013.

[5] P. Hallgren, M. Ochoa, and A. Sabelfeld, "Innercircle: A parallelizable decentralized privacy-preserving location proximity protocol," in *Privacy, Security and Trust (PST)*, 2015.

[6] ——, "Maxpace: Speed-constrained location queries," in *Communications and Network Security (CNS)*, 2016.

[7] S. Stirbys, O. A. Nabah, P. Hallgren, and A. Sabelfeld, "Privacy-preserving location-proximity for mobile apps," in *Parallel, Distributed and Network-based Processing (PDP)*, 2017.

[8] P. Hallgren, C. Orlandi, and A. Sabelfeld, "PrivatePool: Privacy-preserving ridesharing," in *Computer Security Foundations Symposium (CSF)*, 2017.

[9] K. Järvinen, Á. Kiss, T. Schneider, O. Tkachenko, and Z. Yang, "Faster privacy-preserving location proximity schemes," in *CANS*, 2018.

[10] J. Einsiedler, I. Radusch, and K. Wolter, "Vehicle indoor positioning: A survey," in *IEEE Workshop on Positioning, Navigation and Communications (WPNC)*, 2017.

[11] L. Chen, S. Thombre, K. Järvinen, E. S. Lohan, A. Alén-Savikko, H. Leppäkoski, M. Z. H. Bhuiyan, S. Bu-Pasha, G. N. Ferrara, S. Honkala, J. Lindqvist, L. Ruotsalainen, P. Korpisaari, and H. Kuusniemi, "Robustness, security and privacy in location-based services for future IoT: A survey," *IEEE Access*, 2017.

[12] E. S. Lohan, P. Richter, V. Lucas-Sabola, J. A. López-Salcedo, G. Seco-Granados, H. Leppäkoski, and E. S. Santiago, "Location privacy challenges and solutions - parts I and II," *Inside GNSS*, 2017.

[13] A. M. Ladd, K. E. Bekris, G. Marceau, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Using wireless Ethernet for localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2002.

[14] P. Tao, A. Rudys, A. M. Ladd, and D. S. Wallach, "Wireless LAN location-sensing for security applications," in *WiSec*, 2003.

[15] A. Haeberlen, E. Flannery, A. M. Ladd, A. Rudys, D. S. Wallach, and L. E. Kavraki, "Practical robust localization over large-scale 802.11 wireless networks," in *MobiCom*, 2004.

[16] A. M. Ladd, K. E. Bekris, A. Rudys, L. E. Kavraki, and D. S. Wallach, "Robotics-based location sensing using wireless Ethernet," *Wireless Networks*, 2005.

[17] X.-Y. Lin, T.-W. Ho, C.-C. Fang, Z.-S. Yen, B.-J. Yang, and F. Lai, "A mobile indoor positioning system based on ibeacon technology," in *Engineering in Medicine and Biology Society (EMBC)*, 2015.

[18] C. Langlois, S. Tiku, and S. Pasricha, "Indoor localization with smartphones: Harnessing the sensor suite in your pocket," *IEEE CEM*, 2017.

[19] A. Yassin, Y. Nasser, M. Awad, A. Al-Dubai, R. Liu, C. Yuen, R. Raulefs, and E. Aboutanios, "Recent advances in indoor localization: A survey on theoretical approaches and applications," *IEEE Communications Surveys and Tutorials*, 2017.

[20] S. Capkun, S. Ganeriwal, F. Anjum, and M. Srivastava, "Secure RSS-based localization in sensor networks," *Technical Report/ETH Zurich*, 2011.

[21] C. K. Chung, I. Q. Chung, Y. H. Wang, and C. T. Chang, "The integrated applications of WIFI and APP used in the shopping mall environment for menber card E-marketing," in *International Conference on Machine Learning and Computing*, 2016.

[22] T. Guan, L. Fang, W. Dong, Y. Hou, and C. Qiao, "Indoor localization with asymmetric grid-based filters in large areas utilizing smartphones," in *IEEE International Conference on Communications (ICC)*, 2017.

[23] Z. Yin, C. Wu, Z. Yang, and Y. Liu, "Peer-to-peer indoor navigation using smartphones," *IEEE Journal on Selected Areas in Communications*, 2017.

[24] M. Ibrahim, H. Liu, M. Jawahar, V. Nguyen, M. Gruteser, R. Howard, B. Yu, and F. Bai, "Verification: Accuracy evaluation of WiFi fine time measurements on an open platform," in *Mobile Computing and Networking*, 2018.

[25] S. Sen, J. Lee, K.-H. Kim, and P. Congdon, "Avoiding multipath to revive inbuilding WiFi localization," in *Mobile Systems, Applications, and Services*, 2013.

[26] W. Gong and J. Liu, "SiFi: Pushing the limit of time-based WiFi localization using a single commodity access point," *Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2018.

[27] Z. Zhang, X. Zhou, W. Zhang, Y. Zhang, G. Wang, B. Y. Zhao, and H. Zheng, "I am the antenna: Accurate outdoor AP location using smartphones," in *MobiCom*, 2011.

[28] P. Richter and M. Toledano-Ayala, "Revisiting gaussian process regression modeling for localization in wireless sensor networks," *Sensors*, 2015.

[29] K. Sheng, Z. Gu, X. Mao, X. Tian, W. Wu, X. Gan, and X. Wang, "The collocation of measurement points in large open indoor environment," in *INFOCOM*, 2015.

[30] M. Wang, Z. Zhang, X. Tian, and X. Wang, "Temporal correlation of the RSS improves accuracy of fingerprinting localization," in *INFOCOM*, 2016.

[31] G. Bielsa, J. Palacios, A. Loch, D. Steinmetzer, P. Casari, and J. Widmer, "Indoor localization using commercial off-the-shelf 60 GHz access points," in *INFOCOM*, 2018.

[32] S. He, W. Lin, and S.-H. G. Chan, "Indoor localization and automatic fingerprint update with altered AP signals," *IEEE Transactions on Mobile Computing*, 2017.

[33] C. Chen, Y. Chen, Y. Han, H. Q. Lai, F. Zhang, and K. J. R. Liu, "Achieving centimeter-accuracy indoor localization on WiFi platforms: A multi-antenna approach," *IEEE IoT Journal*, 2017.

[34] X. Li, T. Zhang, D. Ma, B. Cao, and Q. Zhang, "Location privacy protection in asynchronous localization networks by resource allocation approaches," in *IEEE International Conference on Communications (ICC) Workshops*, 2017.

[35] Z. Yang and K. Järvinen, "The death and rebirth of privacy-preserving Wifi fingerprint localization with Paillier encryption," in *INFOCOM*, 2018.

[36] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *EUROCRYPT*, 1999.

[37] H. Li, L. Sun, H. Zhu, X. Lu, and X. Cheng, "Achieving privacy preservation in WiFi fingerprint-based localization," in *INFOCOM*, 2014.

[38] T. Shu, Y. Chen, J. Yang, and A. Williams, "Multi-lateral privacy-preserving localization in pervasive environments," in *INFOCOM*, 2014.

[39] J. H. Ziegeldorf, N. Viol, M. Henze, and K. Wehrle, "Poster: Privacy-preserving indoor localization," *WiSec*, 2014.

[40] A. Konstantinidis, G. Chatzimilioudis, D. Zeinalipour-Yazti, P. Mpeis, N. Pelekis, and Y. Theodoridis, "Privacy-preserving indoor localization on smartphones," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2015.

[41] L. Sweeney, "k-anonymity: A model for protecting privacy," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUFKS)*, 2002.

[42] T. Zhang, S. S. M. Chow, Z. Zhou, and M. Li, "Privacy-preserving Wi-Fi fingerprinting indoor localization," in *IWSEC*, 2016.

[43] Y. Huang, P. Chapman, and D. Evans, "Privacy-preserving applications on smartphones," in *HotSec*, 2011.

[44] H. Carter, C. Lever, and P. Traynor, "Whitewash: Outsourcing garbled circuit generation for mobile devices," in *ACSAC*, 2014.

[45] H. Carter, B. Mood, P. Traynor, and K. Butler, "Secure outsourced garbled circuit evaluation for mobile devices," *Journal of Computer Security*, 2016.

[46] P. Richter and M. Toledano-Ayala, "Ubiquitous and seamless localization: fusing GNSS pseudoranges and WLAN signal strengths," *Mobile Information Systems*, 2017.

[47] J. Torres-Sospedra, R. Montoliu, S. Trilles, Ó. Belmonte, and J. Huerta, "Comprehensive analysis of distance and similarity measures for Wi-Fi fingerprinting indoor positioning systems," *Expert Systems with Applications*, 2015.

[48] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of wireless indoor positioning techniques and systems," *Systems, Man, and Cybernetics*, 2007.

[49] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *INFOCOM*, 2000.

[50] B. Li, Y. Wang, H. K. Lee, A. Dempster, and C. Rizos, "Method for yielding a database of location fingerprints in WLAN," *IEEE Proceedings - Communications*, 2005.

[51] D. Demmler, T. Schneider, and M. Zohner, "ABY - a framework for efficient mixed-protocol secure two-party computation," in *NDSS*, 2015.

[52] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits," in *USENIX Security*, 2011.

[53] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi, "ObliVM: A programming framework for secure computation," in *S&P*, 2015.

[54] B. Mood, D. Gupta, H. Carter, K. Butler, and P. Traynor, "Frigate: A validated, extensible, and efficient compiler and interpreter for secure computation," in *EuroS&P*, 2016.

[55] A. Yao, "How to generate and exchange secrets," in *FOCS*, 1986.

[56] O. Goldreich, S. Micali, and A. Wigderson, "How to play any mental game, or a completeness theorem for protocols with an honest majority," in *STOC*, 1987.

[57] D. Beaver, "Efficient multiparty protocols using circuit randomization," in *CRYPTO*, 1991.

[58] Y. Ishai, J. Kilian, K. Nissim, and E. Petrank, "Extending oblivious transfers efficiently," in *CRYPTO*, 2003.

[59] G. Asharov, Y. Lindell, T. Schneider, and M. Zohner, "More efficient oblivious transfer and extensions for faster secure computation," in *CCS*, 2013.

[60] T. Schneider and M. Zohner, "GMW vs. Yao? Efficient secure two-party computation with low depth circuits," in *FC*, 2013.

[61] D. Demmler, G. Dessouky, F. Koushanfar, A.-R. Sadeghi, T. Schneider, and S. Zeitouni, "Automated synthesis of optimized circuits for secure computation," in *CCS*, 2015.

[62] V. Kolesnikov and R. Kumaresan, "Improved OT extension for transferring short secrets," in *CRYPTO*, 2013.

[63] G. Dessouky, F. Koushanfar, A.-R. Sadeghi, T. Schneider, S. Zeitouni, and M. Zohner, "Pushing the communication barrier in secure computation using lookup tables," in *NDSS*, 2017.

[64] D. Malkhi, N. Nisan, B. Pinkas, and Y. Sella, "Fairplay – a secure two-party computation system," in *USENIX Security*, 2004.

[65] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *ICALP*, 2008.

[66] M. Bellare, V. T. Hoang, S. Keelveedhi, and P. Rogaway, "Efficient garbling from a fixed-key blockcipher," in *S&P*, 2013.

[67] S. Zahur, M. Rosulek, and D. Evans, "Two halves make a whole - reducing data transfer in garbled circuits using half gates," in *EUROCRYPT*, 2015.

[68] D. Demmler, T. Schneider, and M. Zohner, "Ad-hoc secure two-party computation on mobile devices using hardware tokens," in *USENIX Security*, 2014.

[69] D. Gupta, B. Mood, J. Feigenbaum, K. Butler, and P. Traynor, "Using Intel software guard extensions for efficient two-party secure function evaluation," in *FC*, 2016.

[70] A. Gascón, P. Schoppmann, B. Balle, M. Raykova, J. Doerner, S. Zahur, and D. Evans, "Privacy-preserving distributed linear regression on high-dimensional data," *PETS*, 2017.

[71] P. Mohassel and Y. Zhang, "SecureML: A system for scalable privacy-preserving machine learning," in *S&P*, 2017.

[72] D. Demmler, K. Hamacher, T. Schneider, and S. Stammler, "Privacy-preserving whole-genome variant queries," in *CANS*, 2017.

[73] C. Bonte, E. Makri, A. Ardeshirdavani, J. Simm, Y. Moreau, and F. Vercauteren, "Towards practical privacy-preserving genome-wide association study," *BMC bioinformatics*, 2018.

[74] O. Tkachenko, C. Weinert, T. Schneider, and K. Hamacher, "Large-scale privacy-preserving statistical computations for distributed genome-wide association studies," in *ASIACCS*, 2018.

[75] T. Schneider and O. Tkachenko, "Towards efficient privacy-preserving similar sequence queries on outsourced genomic databases," in *WPES*, 2018.

[76] H. Cho, D. J. Wu, and B. Berger, "Secure genome-wide association analysis using multiparty computation," *Nature biotechnology*, 2018.

[77] T. Schneider and O. Tkachenko, "EPISODE: Efficient Privacy-preservIng Similar sequence queries on Outsourced genomic DatabasEs," in *ASIACCS*, 2019, to appear.

[78] D. W. Archer, D. Bogdanov, Y. Lindell, L. Kamm, K. Nielsen, J. I. Pagter, N. P. Smart, and R. N. Wright, "From keys to databases—real-world applications of secure multi-party computation," *The Computer Journal*, 2018.
[79] T. Araki, J. Furukawa, Y. Lindell, A. N., and K. Ohara, "High-throughput semi-honest secure three-party computation with an honest majority," in *CCS*, 2016.
[80] P. Mohassel and P. Rindal, "ABY3: A mixed protocol framework for machine learning," in *CCS*, 2018.
[81] T. P. Jakobsen, J. B. Nielsen, and C. Orlandi, "A framework for outsourcing of secure computation," in *CCSW*, 2014.
[82] H. Carter, B. Mood, P. Traynor, and K. Butler, "Outsourcing secure two-party computation as a black box," in *CANS*, 2015.
[83] D. Rotaru and T. Wood, "MArBled circuits: Mixing arithmetic and boolean circuits with active security," Cryptology ePrint Archive, Report 2019/207, 2019, https://eprint.iacr.org/2019/207.
[84] A. B. M. Musa and J. Eriksson, "Tracking unmodified smartphones using Wi-Fi monitors," in *Embedded Network Sensor Systems*, 2012.
[85] M. Vanhoef, C. Matte, M. Cunche, L. S. Cardoso, and F. Piessens, "Why MAC address randomization is not enough: An analysis of Wi-Fi network discovery mechanisms," in *ASIACCS*, 2016.
[86] J. Martin, T. Mayberry, C. Donahue, L. Foppe, L. Brown, C. Riggins, E. C. Rye, and D. Brown, "A study of MAC address randomization in mobile devices and when it fails," *PETS*, 2017.
[87] L. Schauer, F. Dorfmeister, and F. Wirth, "Analyzing passive Wi-Fi fingerprinting for privacy-preserving indoor-positioning," in *International Conference on Localization and GNSS (ICL-GNSS)*, 2016.
[88] S. Kamara and M. Raykova, "Secure outsourced computation in a multi-tenant cloud," in *IBM Workshop on Cryptography and Security in Clouds*, 2011.
[89] A. Gersho and R. M. Gray, *Vector quantization and signal compression*. Kluwer Academic Publishers, 1992.
[90] P. Richter, E. S. Lohan, and J. Talvitie, "WLAN (WiFi) RSS database for fingerprinting positioning," 2018, https://zenodo.org/record/1161525.
[91] P. Richter, H. Leppäkoski, E. S. Lohan, Z. Yang, K. Järvinen, O. Tkachenko, and T. Schneider, "Received signal strength quantization for secure indoor positioning via fingerprinting," in *International Conference on Localization and GNSS (ICL-GNSS)*, 2018.
[92] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider, "Improved garbled circuit building blocks and applications to auctions and computing minima," in *CANS*, 2009.
[93] N. Gilboa, "Two party RSA key generation," in *CRYPTO*, 1999.
[94] J. von Neumann, "First draft of a report on the edvac," *IEEE Annals of the History of Computing*, 1993.
[95] J. Boyar and R. Peralta, "Tight bounds for the multiplicative complexity of symmetric functions," *Theoretical Computer Science*, 2008.
[96] J. Bringer, H. Chabanne, M. Favre, A. Patey, T. Schneider, and M. Zohner, "GSHADE: Faster privacy-preserving distance computation and biometric identification," in *IH&MMSec*, 2014.
[97] E. M. Songhori, S. U. Hussain, A.-R. Sadeghi, and F. Koushanfar, "Compacting privacy-preserving k-nearest neighbor search using logic synthesis," in *DAC*, 2015.
[98] K. E. Batcher, "Sorting networks and their applications," in *Spring Joint Computer Conference*, 1968.
[99] H. W. Lang, "Bitonic sorting network for n not a power of 2," http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/bitonic/oddn.htm, 2017, accessed: 2018-01-16.
[100] M. Keller and P. Scholl, "Efficient, oblivious data structures for MPC," in *ASIACRYPT*, 2014.
[101] N. Buescher, A. Holzer, A. Weber, and S. Katzenbeisser, "Compiling low depth circuits for practical secure computation," in *ESORICS*, 2016.
[102] R. Kohavi and G. H. John, "Wrappers for feature subset selection," *Artificial Intelligence*, 1997.
[103] E. Sansano Sansano, R. Montoliu Colás, O. Belmonte Fernández, and J. Torres-Sospedra, "IndoorLoc Platform, a public repository for comparing and evaluating indoor positioning and navigation databases and algorithms," http://indoorlocplatform.uji.es/, 2017, accessed: 2018-05-07.
[104] D. Dheeru and E. Karra Taniskidou, "UCI machine learning repository," https://archive.ics.uci.edu/ml/, 2017, accessed: 2018-05-07.
[105] D. Kotz, T. Henderson, I. Abyzov, and J. Yeo, "CRAWDAD, a community resource for archiving wireless data at Dartmouth," https://crawdad.org/index.html, 2009, accessed: 2018-05-07.
[106] E. Barker, "Recommendation for key management part 1: General (revision 4)," *NIST*, 2016.

APPENDIX

A. Algorithms

In this appendix, we give details on our algorithm implementations in the ABY framework [51].

$distances \leftarrow d_m^{ABOY}(RSS_C, RefPoints, N)$

1: $distances \leftarrow \emptyset$
2: **foreach** RSS_S **in** $RefPoints$ **do**
3: $\quad\langle dist\rangle^A \leftarrow 0$
4: $\quad$**for** $i = 1 : N$ **do**
5: $\quad\quad\langle gt\rangle^B \leftarrow$ **Y2B**(**GT**($\langle RSS_S[i]\rangle^Y, \langle RSS_C[i]\rangle^Y$))
6: $\quad\quad\langle d\rangle^A \leftarrow \langle RSS_S - RSS_C\rangle^A$
7: $\quad\quad\langle d'\rangle^A \leftarrow$ **Two'sComplement**($\langle d\rangle^A$)
8: $\quad\quad\langle d\rangle^A \leftarrow \langle d\rangle^A \cdot \langle gt\rangle^B$ //using C-OT
9: $\quad\quad\langle d'\rangle^A \leftarrow \langle d'\rangle^A \cdot$ **INV**($\langle gt\rangle^B$) //using C-OT
10: $\quad\quad\langle dist\rangle^A \leftarrow \langle dist\rangle^A + \langle d\rangle^A + \langle d'\rangle^A$
11: $\quad distances$.**append**($\langle dist\rangle^A$)
12: **return** $distances$

Algorithm 1: Manhattan distance using Arithmetic, Boolean, and Yao sharing, and Correlated OT (C-OT).

$distances \leftarrow d_e^A(RSS_C, \langle RSS_C^2\rangle^A, RefPoints, N)$

1: $distances \leftarrow \emptyset$
2: **foreach** RSS_S **in** $RefPoints$ **do**
3: $\quad\langle dist\rangle^A \leftarrow \langle RSS_C^2\rangle^A + \langle RSS_S^2\rangle^A$ //precomputed
4: $\quad$**for** $i = 1 : N$ **do**
5: $\quad\quad\langle product\rangle^A \leftarrow \langle RSS_C[i]\rangle^A \cdot \langle RSS_S[i]\rangle^A$
6: $\quad\quad\langle product\rangle^A \leftarrow \langle product\rangle^A + \langle product\rangle^A$
7: $\quad\quad\langle dist\rangle^A \leftarrow \langle dist\rangle^A - \langle product\rangle^A$
8: $\quad distances$.**append**($\langle dist\rangle^A$)
9: **return** $distances$

Algorithm 2: Euclidean distance using Arithmetic sharing.

$distances \leftarrow d_{kh}^{AY}(RSS_C, \langle RSS_C^2\rangle^A, RefPoints, N, ScalingFactor)$

1: $distances \leftarrow \emptyset$
2: **foreach** RSS_S **in** $RefPoints$ **do**
3: $\quad\langle nom\rangle^A \leftarrow 0$
4: $\quad\langle denom\rangle^A \leftarrow \langle RSS_C^2\rangle^A + \langle RSS_S^2\rangle^A$ //precomputed
5: $\quad$**for** $i = 1 : N$ **do**
6: $\quad\quad\langle product\rangle^A \leftarrow \langle RSS_S[i]\rangle^A \cdot \langle RSS[i]_C\rangle^A$
7: $\quad\quad\langle nom\rangle^A \leftarrow \langle product\rangle^A$
8: $\quad\quad\langle denom\rangle^A \leftarrow \langle denom\rangle^A - \langle product\rangle^A$
9: $\quad\langle nom\rangle^Y \leftarrow$ **A2Y**($\langle nom\rangle^A$)
10: $\quad\langle nom\rangle^Y \leftarrow \langle nom\rangle^Y \ll ScalingFactor$ //bit-shift
11: $\quad\langle denom\rangle^Y \leftarrow$ **A2Y**($\langle denom\rangle^A$)
12: $\quad distances$.**append**($\langle nom\rangle^Y / \langle nom\rangle^Y$)
13: **return** $distances$

Algorithm 3: Kumar-Hassebrook distance using Arithmetic sharing and Yao sharing.

$distances \leftarrow d'^{AY}_s(RSS_C, RefPoints, N, \langle ScalingFactor \rangle^A)$

```
1 :  distances ← ∅
2 :  foreach RSS_S in RefPoints do
3 :      ⟨num⟩^A ← 0
4 :      ⟨denom⟩^A ← 0
5 :      for i = 1 : N do
6 :          ⟨eucl⟩^A ← d_e^A(⟨RSS_S[i]⟩^A, ⟨RSS_C[i]⟩^A)
7 :          ⟨sum⟩^A ← ⟨RSS_S[i]⟩^A + ⟨RSS_C[i]⟩^A
8 :          ⟨num⟩^A ← ⟨num⟩^A + ⟨eucl⟩^A
9 :          ⟨denom⟩^A ← ⟨denom⟩^A + ⟨sum⟩^A
10 :     ⟨num⟩^Y ← A2Y(⟨num⟩^A)
11 :     ⟨num⟩^Y ← ⟨num⟩^Y ≪ ScalingFactor        //bit-shift
12 :     ⟨denom⟩^Y ← A2Y(⟨denom⟩^A)
13 :     distances.append(⟨num⟩^Y / ⟨denom⟩^Y)
14 : return distances
```

Algorithm 4: Modified Sorensen distance using Arithmetic sharing and Yao sharing. Euclidean distance is computed similar to Algorithm 2.

BitonicSort$(dists, ids, start, M)$

```
1 :  for i = 1 : M do
2 :      m = M/2
3 :      BitonicSort(dists, ids, start, m)
4 :      BitonicSort(dists, ids, start + m, M − m)
5 :      BitonicMerge(dists, ids, start, M)
```

BitonicMerge$(dists, ids, start, M)$

```
1 :  m = PO2LT(M)
2 :  for i = begin : begin − M + m
3 :      ⟨gt⟩^B = GT(⟨dists[i]⟩^B, ⟨dists[i + m]⟩^B)
4 :      CondSwap(⟨dists[i]⟩^B, ⟨dists[i + m]⟩^B, ⟨gt⟩^B)
5 :      CondSwap(⟨ids[i]⟩^B, ⟨ids[i + m]⟩^B, ⟨gt⟩^B)
6 :  BitonicMerge(dists, ids, start, m)
7 :  BitonicMerge(dists, ids, start + m, M − m)
```

Algorithm 5: Bitonic Sort algorithm using Boolean sharing. The function $y \leftarrow \textbf{PO2LT}(x)$ finds an y such that y is the biggest power of two less than x.

$minIds \leftarrow k\text{-}NN(dists, M, k)$

```
1 :  minDists, minIds ← ∅
2 :  for i = 1 : k + 1 do
3 :      minDists.append(⟨MAX_VALUE⟩^Y)
4 :      minIds.append(⟨0⟩^Y)
5 :  for i = 1 : M do
6 :      minDists[k + 1] ← dists[i]
7 :      minIds[k + 1] ← ⟨i⟩^Y
8 :      for j = k + 1 : 2 do
9 :          ⟨gt⟩^Y ← GT(⟨minDists[j − 1]⟩^Y, ⟨minDists[j]⟩^Y)
10 :         CondSwap(⟨minDists[j]⟩^Y, ⟨minDists[j − 1]⟩^Y, ⟨gt⟩^Y)
11 :         CondSwap(⟨minIds[j]⟩^Y, ⟨minIds[j − 1]⟩^Y, ⟨gt⟩^Y)
12 : return minIds[1 : k]
```

Algorithm 6: k-Nearest Neighbors (k-NN) circuit using Yao sharing.

$\langle coord \rangle^Y \leftarrow \textbf{OA}^s(coords, \langle idx \rangle^Y, M)$

```
1 :  poTwo ← PO2LT(M)
2 :  for i = M − 1 : 2 · poTwo do
3 :      coords.append(NULL)
4 :  i ← 1
5 :  bitNum ← 0
6 :  while (i < 2 · poTwo) do
7 :      ⟨sel⟩^Y ← ⟨idx⟩^Y[bitNum]
8 :      j ← 0
9 :      while (j < coords.size()) do
10 :         if (⟨coords[j]⟩^Y == NULL)
11 :             j ← j + 2 · i
12 :             continue
13 :         else if(⟨coords[j + i]⟩^Y == NULL)
14 :             j ← j + 2 · i
15 :             continue
16 :         else
17 :             ⟨coords[j]⟩^Y ← MUX(
18 :                 ⟨coords[j + i]⟩^Y, ⟨coords[j]⟩^Y, ⟨sel⟩^Y)
19 :         j ← j + 2 · i
20 :     i ← 2 · i
21 :     bitNum ← bitNum + 1
22 : return ⟨res[0]⟩^Y
```

Algorithm 7: Size-optimized Oblivious Array Access (OA) circuit using Yao sharing.

$\langle coord \rangle^B \leftarrow \textbf{OA}^d(coords, ids, \langle idx \rangle^B, M)$

```
1 :  res ← ∅
2 :  for i in 1 : M do
3 :      res.append(EQ(⟨ids[i]⟩^B, ⟨idx⟩^B))
4 :      ⟨res[i]⟩^B ← MUX(⟨0⟩^B, ⟨coords[i]⟩^B, ⟨res[i]⟩^B)
5 :  for i in 2 : M do
6 :      ⟨res[0]⟩^B ← ⟨res[0]⟩^B ⊕ ⟨res[i]⟩^B
7 :  return ⟨res[0]⟩^B
```

Algorithm 8: Depth-optimized Oblivious Array Access (OA) circuit using Boolean sharing.